第5章 GNUツールを使ったADuC7026開発

―― GCCによる開発とInsightによるデバッグ

山際伸一

付属基板の活用に不可欠なソフトウェア開発ツールのうち，第４章で解説した統合開発環境ではコード・サイズに制限がある．一方，GCCベースの設計ではこうした制限なく開発できる．ここでは，付属CD-ROMからツールをインストールして，ADuC7026向けのソフトウェア開発を行う手順を，GCCを使ったことのない人でもわかるように詳細に解説する．また，GUIを持つデバッガInsightの使いかたを説明する．（編集部）

ここでは，Windows環境によるGNUツール群を活用した組み込みソフトウェア開発を行います．GNUツール群を動作させるためのCygwinをセットアップし，GCCを使ってソフトウェア開発を行います．また，Insightを使ってデバッグを行います．

1．Cygwinのセットアップ

まず，GNUプログラムと親和性の高いCygwinをインストールします．Cygwinとは，UNIXライクなインターフェースをWindows上で与えてくれる環境です．

● Cygwinをインストールする

Cygwinのインストール手順を**図1**に示します．

付属CD-ROMのsoft\gnu\cygwin-1.5.18フォルダにあるsetup.exeを実行します．インストール開始のウィンドウが開くので，［次へ(N)>］ボタンをクリックします(**図1**の①)．

Choose A Download Source画面では，Cygwinのパッケージがどこにあるかを指定します．「Install from Local Directory」をチェックし，［次へ(N)>］ボタンをクリックします(**図1**の②)．

Select Root Install Directory画面では，インストール先を指定します．とくに問題がなければデフォルトのままで［次へ(N)>］ボタンをクリックします(**図1**の③)．

Select Local Package Directory画面では，パッケージの具体的な所在を指定します．［Browse...］ボタンをクリックするとフォルダを参照できます．setup.exeと同じフォルダを指定して，［次へ(N)>］ボタンをクリックします(**図1**の④)．

Select Package画面では，インストールするパッケージ

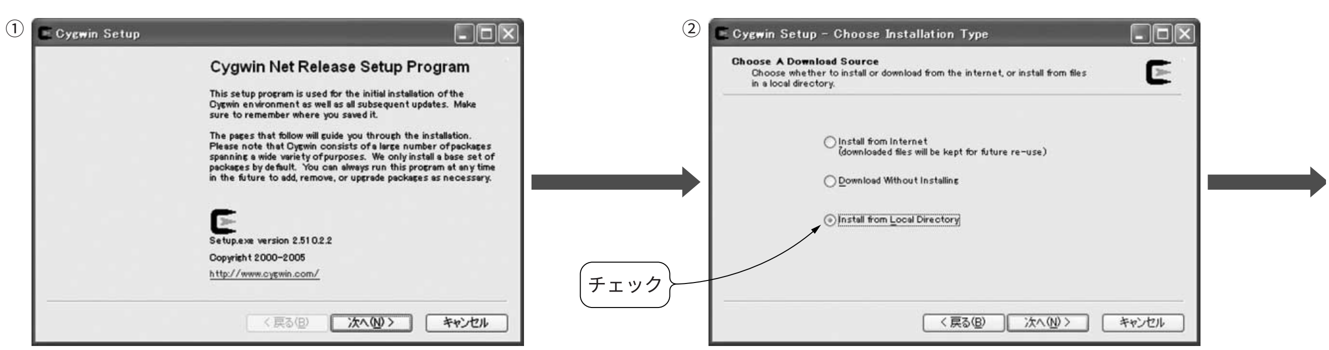


図1　Cygwinのインストール手順

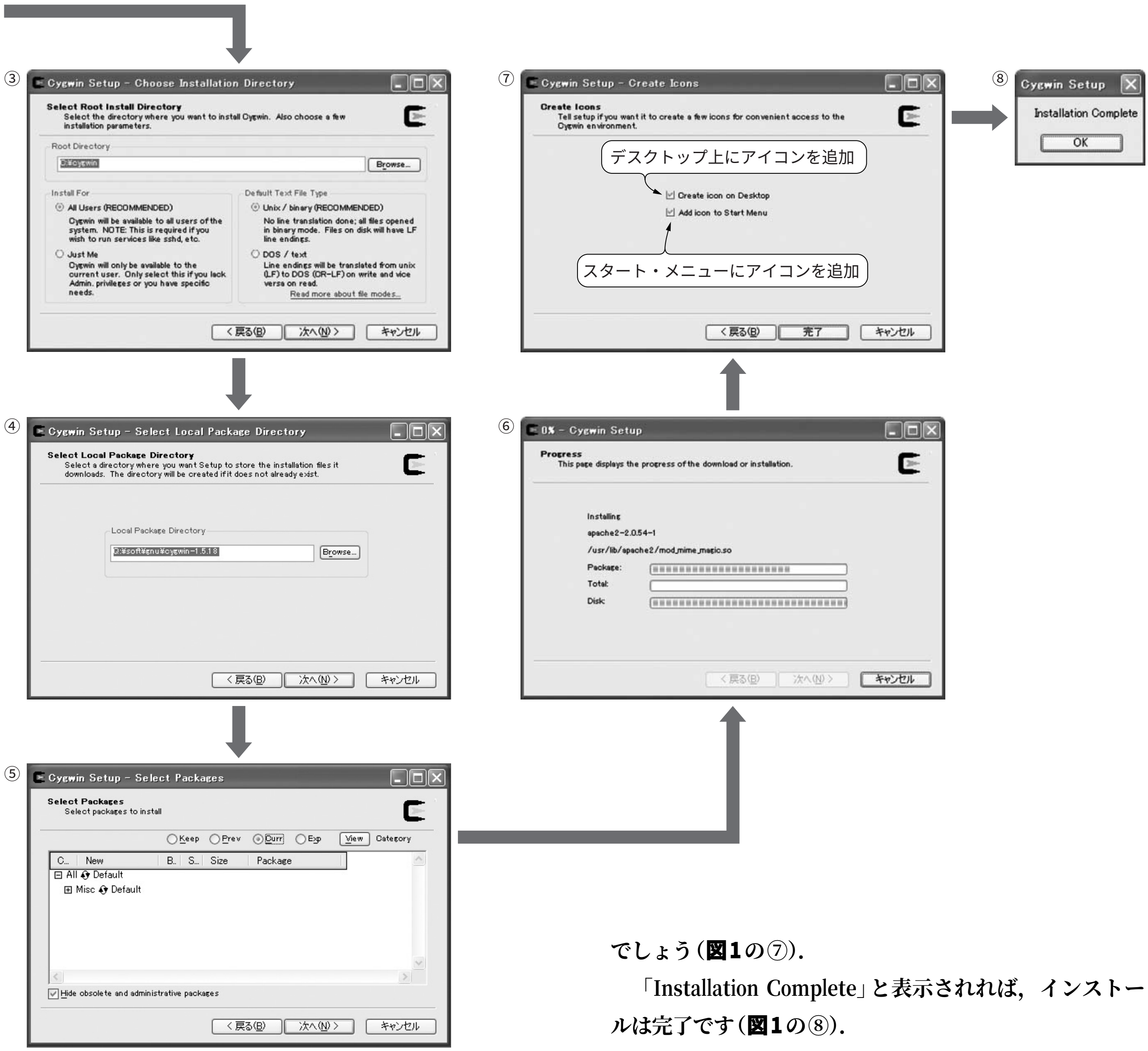


図1　Cygwinのインストール手順（前ページのつづき）

を指定します．GNUツール群についてよくご存じの方で，ほんとうにインストールしたくないと思うパッケージなどがない限り，デフォルトのまま［次へ(N) >］ボタンをクリックしてください（**図1**の⑤）．

これで設定は終わりです．10分くらいでファイルのコピーは完了します（**図1**の⑥）．

インストールが完了すると，Create Icons画面でデスクトップとスタート・メニューにアイコンを追加するかという質問が出ます．アイコンの作成は任意ですが，今後活用していくうえでは，両方チェックして作成したほうがよいでしょう（**図1**の⑦）．

「Installation Complete」と表示されれば，インストールは完了です（**図1**の⑧）．

● コンソールを起動する

Cygwin環境がインストールできたら，Cygwinのコンソールを起動して，動作を確認します．

Windowsの「スタート」メニューから「すべてのプログラム」→「Cygwin」→「Cygwin Bash Shell」を選択（または，デスクトップ上のCygwinアイコンをダブルクリック）してください．すると，**図2**のようにコンソールが開きます．

開いたコンソールで，

```
pwd
```

と入力し，［Enter］キーを押してください．pwdは，現在のディレクトリ・パスを表示させるコマンドです（右掲の

Column Cygwinでよく使うコマンド

Cygwin はUNIXライクのインターフェースをWindows上で与えてくれる環境です．したがって，CygwinではUNIX互換のコマンドが用いられます．

以下によく使うコマンドを列挙します．

● mkdir

ディレクトリを作成するコマンドです． make directoryの意味です．

● cd

引き数に指定したディレクトリに，カレント・ディレクトリを変更します．change directoryの意味です．

例えば，以下のような使い道があります．

ARMというディレクトリを作って，そこに移動

```
$ mkdir ARM
$ cd ARM
```

一つ上のディレクトリに移動

```
$ cd ..
```

● cat

引き数に与えられたファイルをコンソールに表示します．例えば，

```
$ cat hello.c
```

とコンソールに入力すると，hello.cの内容がコンソールに表示されます．

● ls

現在のディレクトリの内容を表示します．オプションとして，「-l」「-a」を組み合わせると，ファイルの容量など詳しい情報がコンソールに表示されます．

● rm

引き数として与えたファイルを削除します．

● cp

ファイルをコピーします．

```
cp コピー元 コピー先
```

というように，二つの引き数を指定します．

● mv

ファイルを移動します．

```
mv 移動元 移動先
```

というように，二つの引き数を指定します．

● less

DOSプロンプトで用いられる`more`コマンドに似た，ファイル内容を表示するコマンドです．引き数として与えたファイルを表示します．終了する際は[q]キーを押してください．

● リダイレクト

コマンドではありませんが，「リダイレクト」という機能があります．これはDOSプロンプトでも用いられる方法で，コマンドやコンソールにおける入力を直接ファイルに流し込むという機能です．

例えば，`ls`コマンドの出力をtmp.txtに保存しておきたい場合は，

```
$ ls > tmp.txt
```

とします．

コラム「Cygwinでよく使うコマンド」を参照）．

```
/home/yama
```

あるいは，

```
/cygdrive/c/Documents and Settings/
                                yama
```

のように表示されます．yamaは筆者のユーザ名なので，みなさんのユーザ名になっていると思います．これを「ホーム・ディレクトリ」と呼びます．Windowsのパスで表現すると，/home/yamaはC:\cygwin\home\yama，/cygdrive/c/Documents and Settings/yamaはC:\Documents and Settings/yamaです．

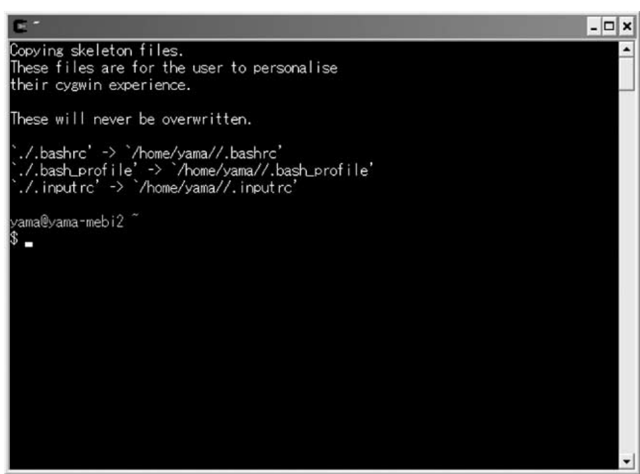


図2 Cygwinのコンソール
Windowsの「スタート」メニューから「すべてのプログラム」→「Cygwin」→「Cygwin Bash Shell」を選択（または，デスクトップ上のCygwinアイコンをダブルクリック）して起動する．

5

Cygwinは，Windows上の環境のため，活用にあたってはWindowsのソフトウェア群と併用することになります．例えば，Windowsのテキスト・エディタを使ってソース・コードを記述したり，Windowsのエクスプローラを使って，Windowsのフォルダからホーム・ディレクトリにファイルのコピーを行うことがあります．ホーム・ディレクトリとWindowsのフォルダの関係をよく覚えておいてください．

● Cプログラムを動作させる

Cygwinの環境に慣れるため，また，「C言語なんて忘れたよ…」という方のために，C言語の入門書でよく用いられる「Hello World」を画面に表示させるプログラムを作ってみましょう．

リスト1　Hello Worldを表示するCソース・コード

```
#include<stdio.h>

int main(){
      printf("Hello World¥n");
      return 0;
}
```

1）ソース・コードの作成

リスト1に，Hello Worldを表示するCソース・コードを示します．ホーム・ディレクトリにhello.cという名まえで作ります．Windowsのエクスプローラでホーム・ディレクトリのフォルダを開いて右クリックし，「新規作成」→「テキストドキュメント」を選択するとメモ帳などのテキスト・エディタが開くはずです．

2）コンパイル

それでは，コンパイルしてみましょう．Cygwinのコンソールで，以下のコマンドを実行してください（$はプロンプトで，画面にあらかじめ表示されている文字である．文字の入力が終わったら最後に[Enter]キーを押す）．

```
$ gcc hello.c -o hello
```

これで，helloというコマンドができました．

3）動作確認

さっそく実行してみましょう．

Column　クロス開発環境をコンパイルする

なんらかの理由でarm-tools.tar.gzを展開したコマンドが動作しない場合があるので，ソース・コードからクロス開発環境を作る方法を説明します．

本誌付属のCD-ROMの中に，arm-tools-src.tar.gzというパッケージを収録しています．このファイルをホーム・ディレクトリにコピーし，次のコマンドを実行します．

```
$ cd ~
$ tar zxvf arm-tools-src.tar.gz
$ ./auto-compile.sh
```

「auto-compile.sh」というスクリプトは，自動的に構築作業を行うためのものです．このスクリプトを**リストA**に示します．クロス環境を構築する際には，binutilsというパッケージ，GCCパッケージという順番で構築する必要があります．さらに，GCCを構築する際には，PATH環境変数にbinutilsで構築したコマンドへのパスを追加しなければなりません．最後にデバッガInsightを構築しています．

約1時間から2時間くらい静かに見守っていると，一連のコンパイルが完了します．

リストA　ARM開発環境自動構築スクリプト

```
#!/bin/sh
tar zxvf binutils-2.16.tar.gz
cd binutils-2.16
./configure --target=arm-elf --prefix=/usr/local/arm-tools
make
make install
cd ..

export PATH=/usr/local/arm-tools/bin:$PATH
tar jxvf gcc-3.4.4.tar.bz2
cd gcc-3.4.4
./configure --target=arm-elf --prefix=/usr/local/arm-tools --enable-languages=c
make
make install
cd ..

tar jxvf insight-weekly-CVS-6.30.50.20051031.tar.bz2
cd src
./configure --target=arm-elf --prefix=/usr/local/arm-tools/gdb
make
make install
cd ..
```

```
$ ./hello
```

画面に，

```
Hello World
```

と表示されたと思います．

2．ARMマイコン向けC言語開発環境

GNUはフリー・ソフトウェアとしてとてもよく知られていますが，その信頼性は高く，米国RedHat社のeCosなどの商用組み込み開発環境においても使われています．しかし，初心者にとって，目的の「組み込みソフトウェアを作成する環境」までを構築するには少々しきいが高いかもしれません．ここでは，ARMマイコン向けの開発環境を構築し，付属基板で動作させるまでの手順を説明します．

● ARM開発環境をインストールする

すでにインストールしているCygwinの環境向けに，あらかじめ構築を完了しているARMマイコン向けのC言語開発環境を本誌付属CD-ROMに収めてあります．

セットアップの手順を以下に示します（左掲のコラム「クロス開発環境をコンパイルする」を参照）．

Windows環境で，付属CD-ROMのsoft\gnuフォルダにあるarm-tools.tar.gzをCygwinのホーム・ディレクトリにコピーします．そしてCygwinのコンソールで，以下の二つのコマンドを順番に実行します．

```
$ cd /usr/local
$ tar zxvf ~/arm-tools.tar.gz
```

これで，Cygwinの/usr/local/arm-toolsに開発環境がインストールされました．

```
$ ls /usr/local/arm-tools/bin
```

と入力すれば，コンパイラやリンカなどのツール群の一覧が表示されます．

● ADuC7026向けソフトウェアを開発する

セットアップしたGNUツールを使って，ADuC7026向けソフトウェアを開発してみます．付属基板のLED（D1）を1秒間隔で点滅させます．

1）ソース・コードの準備

Cソース・コードはやや長くなるため，あらかじめ用意してあるものを使います．付属CD-ROMのarticle\0603_a05\dataにあるgnu_tool_practiceフォルダをCygwinのホーム・ディレクトリにコピーしてください．

今回使うソース・コードは，practice4.cです．テキスト・エディタなどで内容を確認し，ソース・コードを作成したものと思ってください．

ほかにstartup3.sとmemory3.defという二つのファイルも使いますので，存在することを確認しておいてください．

2）オブジェクト・コードの生成

ADuC7026向けのオブジェクト・コードは，以下のコマンドを実行することで生成されます．

```
$ cd gnu_tool_practice
$ /usr/local/arm-tools/bin/
                arm-elf-gcc -c practice4.c
$ /usr/local/arm-tools/bin/
     arm-elf-as startup3.s -o  startup3.o
$ /usr/local/arm-tools/bin/
                arm-elf-ld -T  memory3.def
              startup3.o practice4.o  -Map
                practice4.map -o practice4
$ /usr/local/arm-tools/bin/
         arm-elf-objcopy -O ihex practice4
                              practice4.hex
```

これでオブジェクト・ファイルができ上がりました．米国Analog Devices社のダウンローダARMWSDを使って付属基板に書き込みます（**図3**）．LEDが点滅することを確認できると思います（ARMWSDの操作については，本特

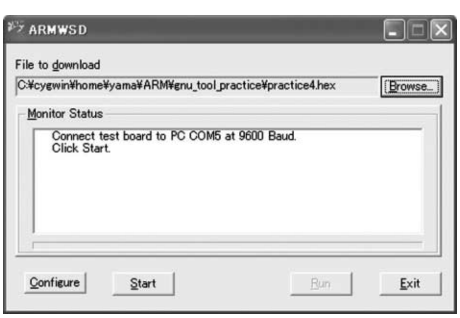


図3　ダウンローダARMWSD
ARMWSDはAnalog Devices社のツールである．使いかたについては，本特集の第4章で解説している．

集の第4章の説明を参照).

3. GUI付きデバッガInsight

GNUフリー・ソフトウェアでは,GDBと呼ばれるデバッガが提供されています.GDBはコマンド・ライン・ベースのデバッガでしたが,RedHat社で使いやすいGUIが構築され,Insightとして世に送り出されました.

GDBはもともと,GDBそのものを実行しているパソコン上で動作するプログラムをデバッグするために開発されたものです.しかし,クロス開発環境にも対応できるように変更が加えられ,最近では組み込み機器開発に用いられています.

Insightは,GDBにGUIをかぶせてあるものなので,GDBの機能はそのまま保持されています.

それではInsightをインストールしましょう…と言いたいところですが,実はすでにインストールされています./usr/local/arm-tools/gdbにInsightがセットアップされています.

試しに,起動してみましょう.

```
$ /usr/local/arm-tools/gdb/bin/
                         arm-elf-insight.exe
```

図4に示すようなウィンドウが起動すると思います.

● デバッグの準備を行う

付属CD-ROMのarticle\0603_a05\dataフォルダの中にあるgdb-stubフォルダをCygwinのホーム・ディレクトリにコピーします.

gdb-stubフォルダの下には,monitorフォルダとdownload_sampleフォルダがあります.monitorフォルダには,Insightと通信してデバッグを進めるための通信プロトコルをもつソフトウェアがあります.download_sampleフォルダには,デバッグ対象となるC言語サンプル・コードが入っています.このサンプルは,LEDを点滅させるものです.

シリアル・ケーブルで付属基板とホストのパソコンを接続します(**図5**).そして,付属基板にmonitorをダウンロードします.ダウンローダARMWSDを起動し,monitor.hexファイルを選択して書き込みます.書き込み完了後は,ARMWSDを閉じて,付属基板のSW1を押してリセットをかけます.モニタ・プログラムが動き出すので,Insightとの通信を待ちます.

● デバッグ対象のコードを準備する

Cygwinコンソールを開いて,「download_sample」ディレクトリに移動します.

```
$ cd ~/gdb-stub/download_sample
```

次にInsightを起動します.

```
$ /usr/local/arm-tools/gdb/bin/
          arm-elf-insight.exe practice2
```

すると**図6**のようなウィンドウが開きます.残念ながら日本語がサポートされていないため,文字化けしますが,

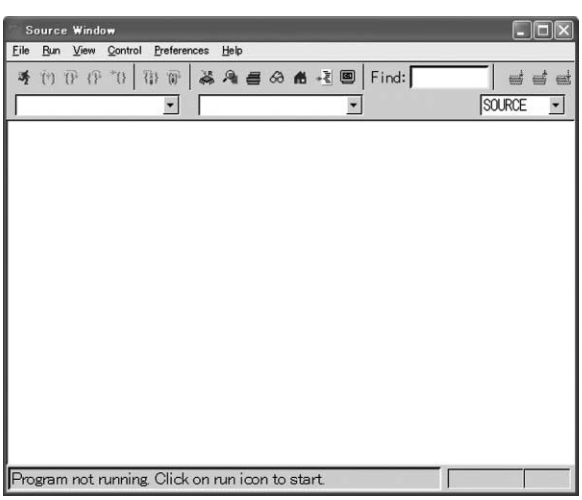


図4　Insightの起動時の画面

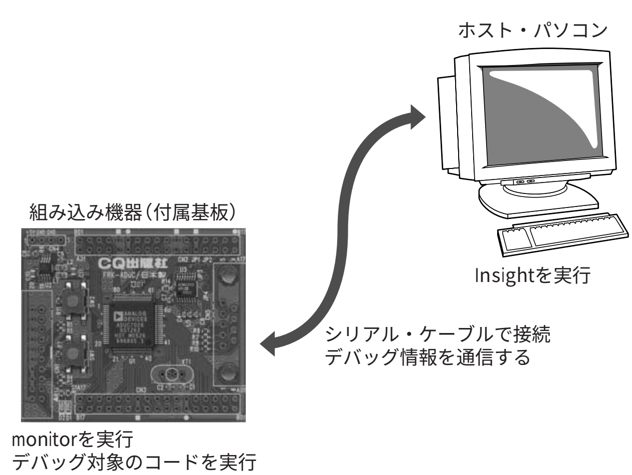


図5　付属基板とホスト・パソコンの接続

ホスト・パソコンでInsightが,付属基板でmonitorとデバッグ対象のコードが動作する.デバッグ情報はシリアル・ケーブルを介して通信する.

ソース・コードと逆アセンブル結果を見ることができます．

これらのコードの情報は，実行形式であるpractice2に埋め込まれています．デバッグ情報を実行形式ファイルに埋め込むためには，以下のようにコンパイラに「-g」オプションを与えてオブジェクト・ファイルを作ります．

```
$ /usr/local/arm-tools/bin/
            arm-elf-gcc -g -c practice2.c
```

また，Insightに与えるファイルは実行形式ファイルであり，Intel HEXファイルでないことに気づいた方もいるでしょう．InsightではELFと呼ばれる実行形式のフォーマットを読み，アドレス構成などの情報を解析して，後述するプログラムのダウンロードなどを行っています．

● デバッグ環境をセットアップする

Insightと付属基板との通信設定や，GDBコンソールを使ってコマンドを与える方法を解説します．

「View」→「Console」を選択します．すると，

```
(gdb)
```

というプロンプトがあるウィンドウが開きます．Insightにはコンソール・ウィンドウのほか，メモリ・ウィンドウやレジスタ・ウィンドウなどがあります．必要に応じて開いてみてください．

ここでは，コンソール・ウィンドウでデバッグ環境のセットアップを行います．

まず，通信中のメッセージをすべてコンソール・ウィンドウに表示させる指定を行います．

```
(gdb) set debug remote 1
```

モニタの通信速度を9600bpsにします．

```
(gdb) set remotebaud 9600
```

GDBリモート・シリアル・プロトコルの拡張モードでCOM1ポートで通信する設定を行います．使用している環境に合わせて，COMポートの番号を変更してください．

```
(gdb) target extended-remote/
                                  dev/com1
```

これで，Insightと付録基板がシリアル・ケーブルを介して接続されました．

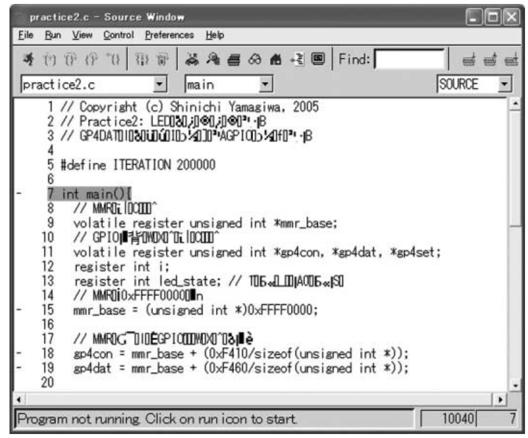


図6　デバッグ対象コードの表示
ソース・コードと逆アセンブル結果が表示される．

● ソフトウェアをダウンロードする

practice2をダウンロードしてみましょう．コンソール・ウィンドウに，

```
(gdb) load
```

と入力してください．実行形式ファイルの中の各セクションが該当するアドレスに書き込まれていきます．

● ブレークポイントを設定する

ここでブレークポイント機能を使ってみます．

ソース・ウィンドウに表示されているソース・コードの左にある「－」というマークの上にマウス・ポインタを持っていき，クリックすると赤い■が出てきます．これでブレークポイントの設定ができました．

ここでは最後のほうのwhile文の中の，

```
led_state = 1;
```

に設定してください．

● プログラムを実行する

ダウンロードしたプログラムを実行するには，コンソール・ウィンドウに，

```
(gdb) run
```

と入力します．このrunコマンドによって，Insightが現在実行しているプログラムを中断するかどうかの確認ダイアログが表示されたら，［Yes］ボタンをクリックします．

通信が始まり，プログラムが実行されます．

5

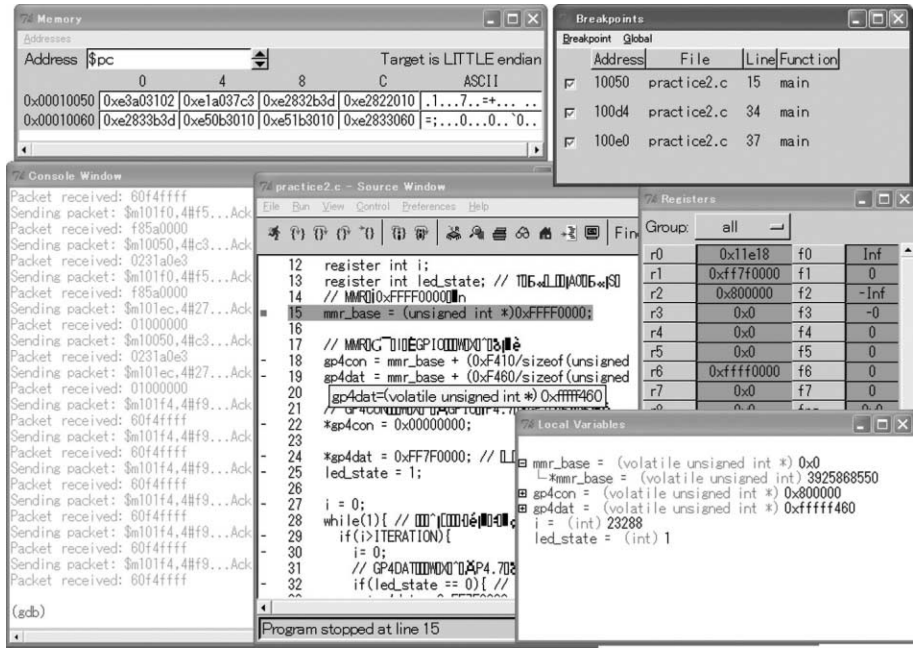


図7
Insightによるデバッグのようす
メモリ・ウィンドウなども開いている状態を示す．

なぜか，Insightでは先ほど設定したブレークポイントではなく，

```
asm("SWI 0x10");
```

という行でInsightに制御が戻ると思います．これは設定した番地とは異なる，と思われるでしょう．実は，これもブレークポイントの一種で，ソフトウェア例外を発生させる命令をC言語プログラムに無理やり置くことで，そのポイントでプログラムを停止することができるのです．こうすることで，事前にプログラム中にブレークポイントを設定することが可能になります．

プログラムを続行するために，

```
(gdb) c
```

とコンソール・ウィンドウに入力します．これで，プログラムは続行し，先ほど設定したブレークポイントに達すると，ソース・ウィンドウのブレークポイントの行にハイライトが移り，Insightに制御が戻ってきます．そして，次のブレークポイントを設定します．

この操作を続けることで，プログラムが正しく動いているかどうかを確認していきます．

さらに，再度ブレークポイントを設定したときには，コンティニュー・コマンドをコンソール・ウィンドウに入力します．

```
(gdb) c
```

runコマンドでないことを注意してください．コンティニュー・コマンドは現在の停止位置からプログラムを続行します．

ブレークポイントに到達した時点の変数の値が見たい場合は，ソース・ウィンドウの変数の上にマウス・ポインタを持っていきます．すると，その変数の内容がポップアップ・ウィンドウに表示されます．

そのほか，メモリ・ウィンドウやさまざまなウィンドウを開いてデバッグが進行しているようすを**図7**に示します．

● Insightを終了する

ソース・ウィンドウを閉じればInsightを終了することができます．開いたウィンドウの状態はホーム・ディレクトリのgdbtk.iniに保存されます．次回，Insightを起動した場合は，このファイルを読み込んでウィンドウ状態などを復帰させます．デフォルト状態が良い場合，このファイルを消してからInsightを起動してください．

やまぎわ・しんいち
工学博士

<筆者プロフィール>
ポルトガルと日本を舞台に，ソフトウェアとハードウェアの開発に飛び回るエンジニア．学会で論文発表も行っている．